All I Ever Needed to Know About Programming

I Learned From Re-writing Classic Arcade Games

Katrin Bec r
J. R. Park

Overview

- The Challenge of reaching students.
- What we usually do in 1st year.
- Games and Students
- Effective Learning
- Games and Pedagogy
- Why Arcade Games
- What can we teach with games?
- Dialing Down & Ramping Up

The Challenge

- Enrollments in CS are down.
- Demand for CS professionals is changing.
  - Need More breadth (applications, media).
  - Outsourcing.
- How to fill demand for programmers?
- How to attract CS Majors?

The Usual

- Typical progression:
  - Uniform, incremental steps
  - Mathematical sequencing of content
  - Teach programming step-wise,
    - From little to big;
    - From simple (sorting lists)...to complex (sorting big lists)
    - From boring to….

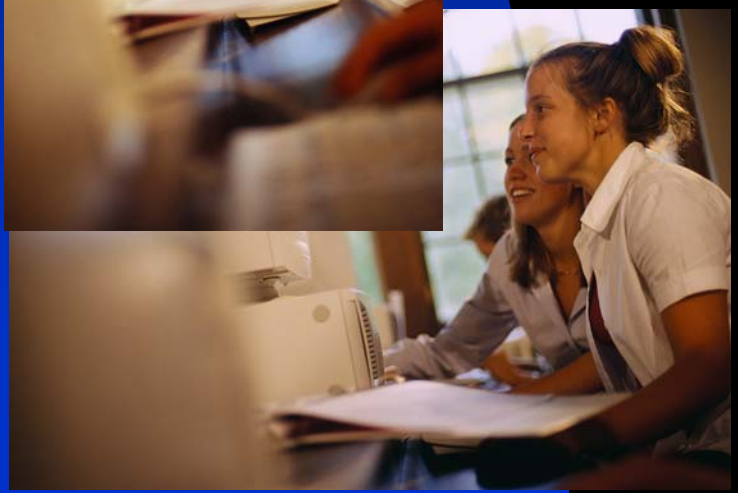
Still boring

Games and Students

- CS students are gamers
- Games got many students interested in CS
- Students *get* games
  - Usually better than “widgets”

Effective learning happens when…

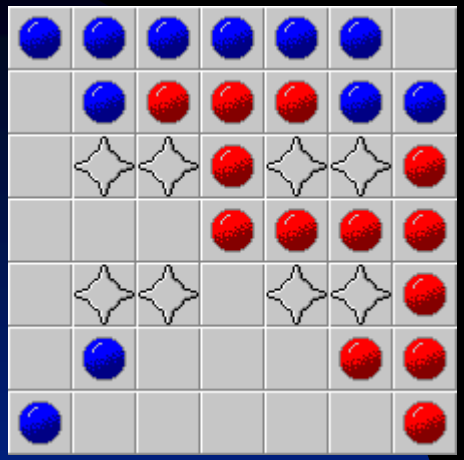
- Students care about the problems they need to solve
- Students understand the problems they need to solve
- We take prior learning into account (start from where the students are)

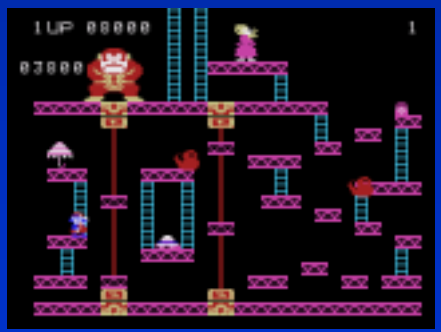


Are Games all this?

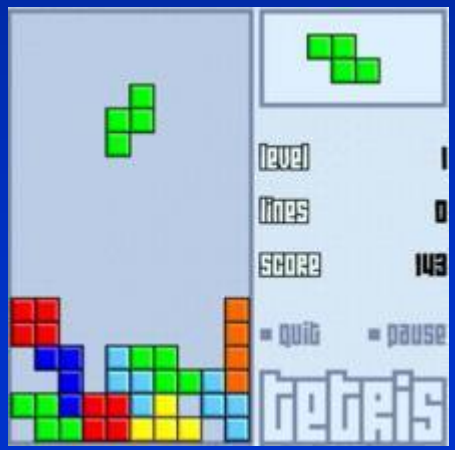
- Students care about games
- Students understand how the games are supposed to work
- Prior knowledge for students includes games literacy.

Games and Pedagogy





- Gameplay is tied to programming:
  - Complex gameplay == complex (and more) algorithms
- Games are highly visual:
  - On-screen behavior maps onto algorithms in-program
  - We can watch the algorithms as they execute
- Program testing techniques are understood and accepted
  - Game cheats
  - Try it and see….(game attitude)



But…

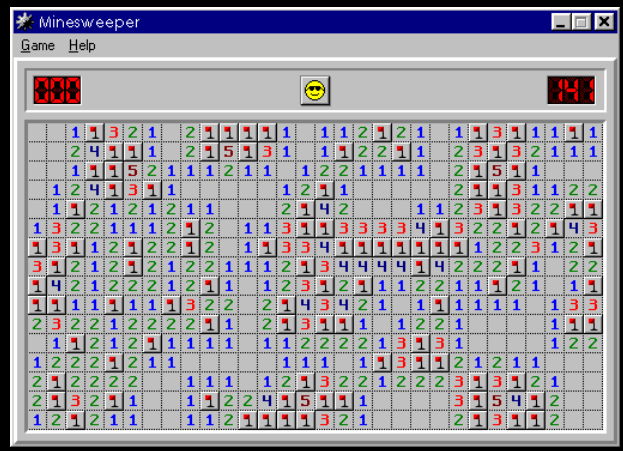
- Games are frivolous…
  - Our goal is to implement the game, not just play it.
- Games are graphics intensive…
  - This doesn't need to be (enter, the arcade).
- Games are primarily event-driven, and that's only one aspect of programming….
  - Most games can be modified to be turn-based.

What's So Special About Arcade Games?

1. Familiarity

- Build something they have actually used (How many freshman have managed enough employees to require a program to keep track of them?)

Go from experiencing the magic..



to being the magician.

What's So Special About Arcade Games?

2. Age

- Built when computers were limited & effects were crude.
  - Program complexity was low
  - Graphics were simple
  - Audio was insignificant (almost)

What's So Special About Arcade Games?

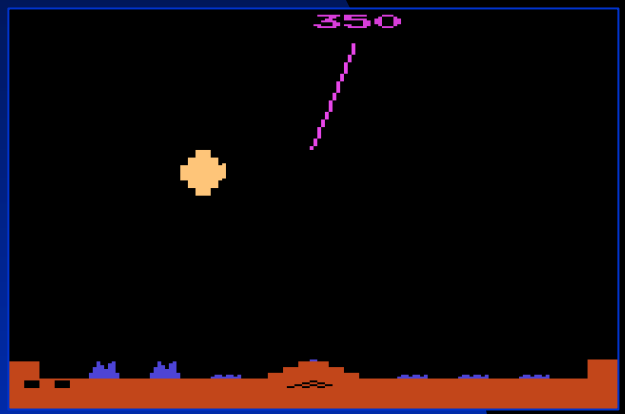
3. Lots of implementations exist.

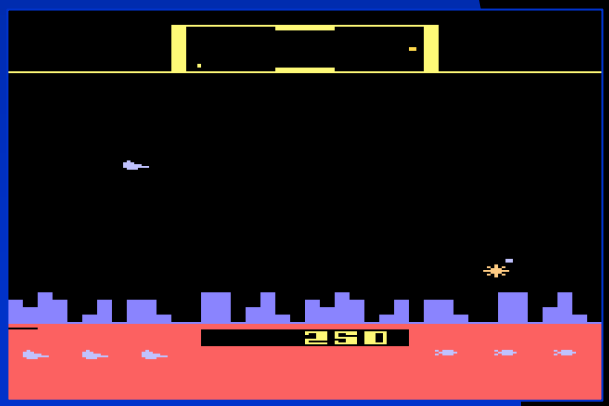
- Working examples are useful when writing a new program.

- Is this a liability? What about cheating?
  - Pick a 1st year programming problem for which no solution exists.

    Anywhere.

    Just try.





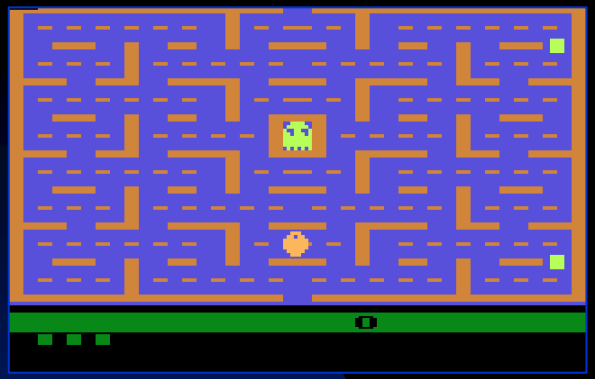
So Many Concepts So Many Games

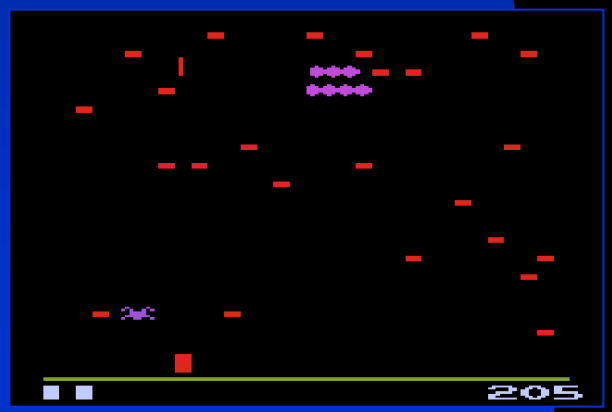
Action Shooters

- Collision detection
- Distance calculations

Asteroids!
Missile Command
Defender

So Many Concepts
So Many Games

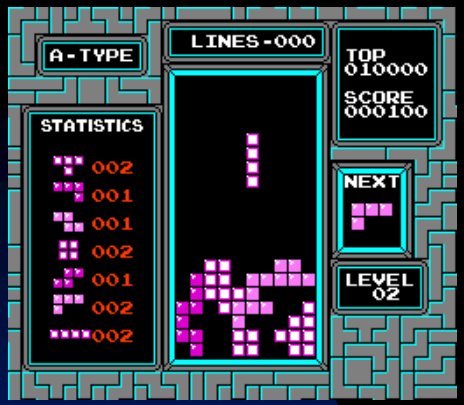




Maze Puzzles

- Path finding
- Chasing (tracking)
- 2D

Pac-Man
Ms. Pac-Man
Centipede



So Many Concepts So Many Games

Puzzles

- 2D geometry
- Packing algorithms
- Win-state detection
- Condition checking

Tetris
Q*Bert

So Many Concepts
So Many Games

Bouncing

- Real-time in a simple environment
- Collision detection
- Simple physics

Blocks
Breakout!
Pong

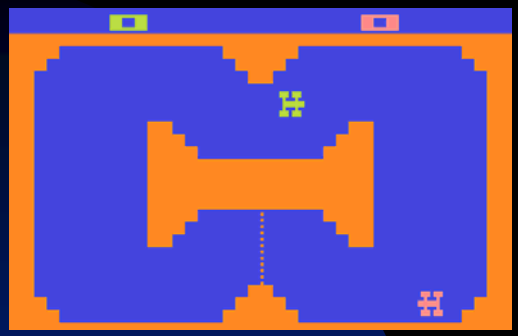
So Many Concepts So Many Games

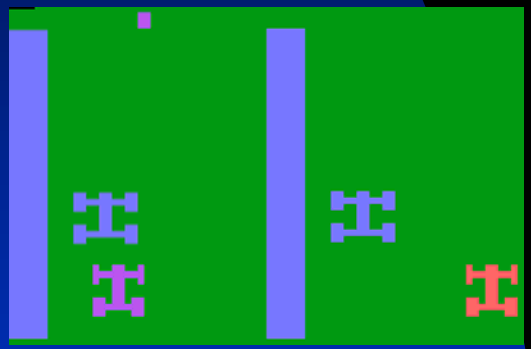
Side Scrollers / Level Games

- Physics
- Inventory & asset management
- AI

Mario Bros.
Donkey Kong
Pitfall
Joust

So Many Concepts
So Many Games





Racing & Driving

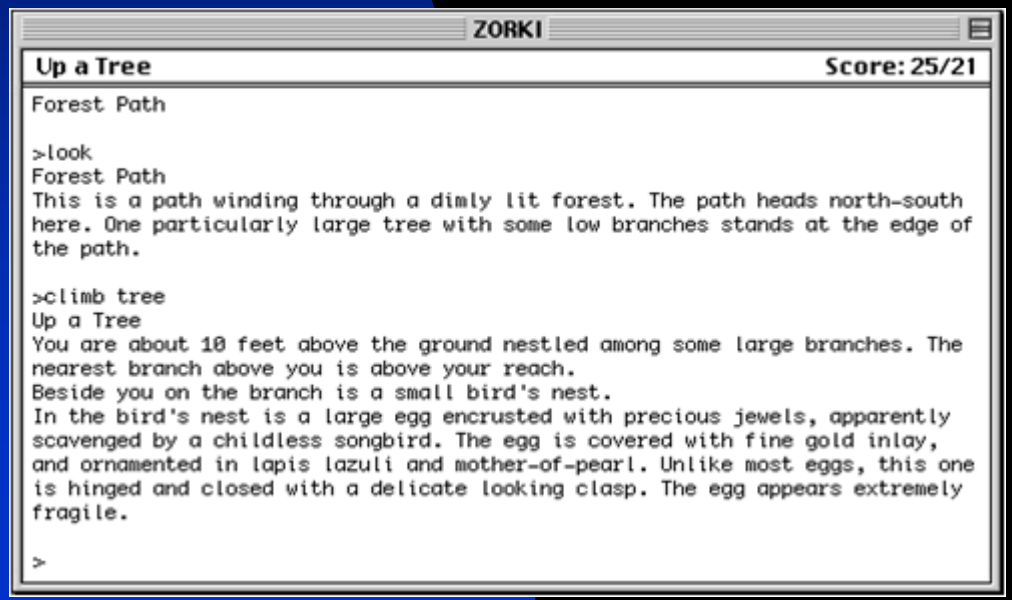
- Physics
- AI
- Collision Detection
- Graphics
- User interfaces
- Audio
- ….

Indy 500
Street Racer

So Many Concepts
So Many Games

Text Based

- Parsing
- AI



Zork

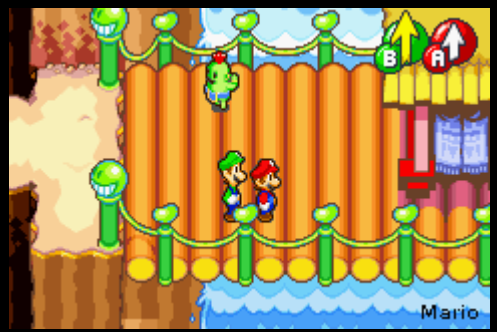
Adding and Removing Complexity

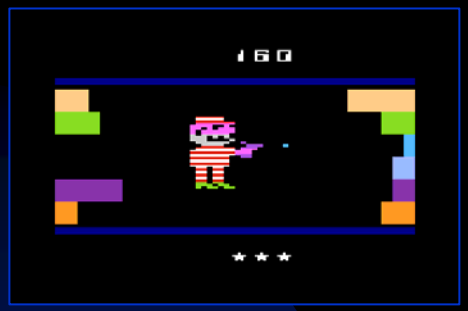
Dialing Down:

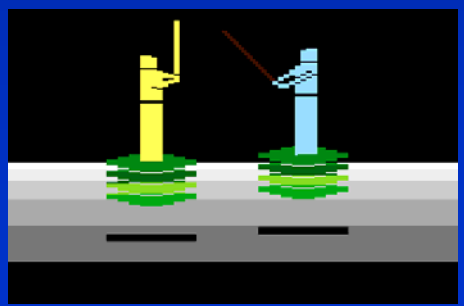
- Can simplify gameplay without loosing appeal
- Can do ASCII games
- Provide plug-ins for tougher bits
- Staged (i.e. C-solution does *this*; B-solution does *this;* and A-solution does *that*)

Ramping Up:

- Can add to even simple games (3D, real-time, full-color, sound,…)
- Concentrate on one aspect – increase quality / complexity for greater challenge (graphics for *Donkey Kong*; physics in driving game; parsing in *Zork*)







Summary

Classic Arcade Games are ideal:

- Complexity at a level novices can master.
- “Special Effects” (graphics, sound, etc.) demands are modest without having to change the original game.
- They are examples of programs with which most students are familiar.
- Many working examples exist and are freely available.
- Complexity and challenge can be easily adjusted to meet requirements for novices and experienced programmers alike.

Thanks.


SCORE
400
052
:00
TIME
BANANA(S)
004 / 100
CLEAR SCORE
TIME BONUS
GOAL!!
7 mph
B
E
FLOOR 1
Super Monkeyball

Image credits

- Attaxx: http://www.pressibus.org/ataxx/gen/gbintro.html
- Breakout: http://freespace.virgin.net/james.handlon/earlygamingmemoriespart2.htm
- Q*Bert: http://www.gamespot.com
- Lunar Lander: http://gnm5.tripod.com/Lunar_lander.htm
- Joust: http://www.glitchnyc.com/cgi-bin/blosxom.cgi/technology/games/index.phblox
- Zork: Infocom, Inc. *Zork I: The Great Underground Empire.* Cambridge, MA: 1981.
- Street Racer: http://www.vgmuseum.com/pics5/streetracer.html
- Pitfall: http://www.geocities.com/xinet2004/segagenesis.htm

- VideoGameCritic.net (multiple images, especially Atari 2600 Games, and Intellivision)
- GameSpot
- MobyGames